Terbit *online* pada laman web jurnal : http://innovatics.unsil.ac.id

Innovation in Research of Informatics (INNOVATICS)

| ISSN (Print) xxx-xxx | ISSN (Online) xxx-xxx |

Websocket untuk Optimasi Kecepatan Data Transfer pada Real Time Chatting

Asep Rizki Maulana[1], Alam Rahmatulloh[2]

[1,2] Teknik Informatika, Fakultas Teknik, Universitas Siliwangi, Indonesia

[1]*pedoelski@gmail.com*, [2]*alam@unsil.ac.id*

INFORMASI ARTIKEL

KATA KUNCI

AJAX,
chatting,
long polling,
real time,
websocket

KORESPONDENSI

Telepon: +6285223519009

E-mail: alam@unsil.ac.id

A B S T R A K

Bentuk komunikasi yang sedang menjadi trending saat ini adalah bertukar pesan atau *chatting*. *Chatting* memanfaatkan teknologi berkirim pesan secara *real time* menggunakan teks melalui jaringan internal. Metode *chatting* dimulai dari *synchronous* sampai sekarang *Asynchronous JavaScript and XML* (*AJAX*). *AJAX* dapat menggunakan metode *polling* ataupun *long polling*. Metode *long polling* memiliki kekurangan karena melakukan proses *request* secara terus menerus. Jika banyak pengguna yang melakukan *request* dapat menyebabkan *server* menjadi sibuk dan *down*. Pada penelitian ini untuk mengatasi permasalahan tersebut akan dibuat aplikasi chatting dengan menggunakan komunikasi full-duplex dengan memanfaatkan teknologi web socket. Hasil percobaan pada penelitian menunjukan bahwa penggunaan *web socket* mampu mengurangi lalu lintas jaringan dan *latency* sehingga lebih baik dari metode AJAX. Nilai presentase transmit data dan receive data dari implementasi AJAX adalah 90,37% dan 94,88%. Setelah metode web socket diterapkan nilai presentase transmit data dan receive masing-masing lebih baik menjadi 9,63% dan 5,12%.

1. PENDAHULUAN

Ditemukannya internet pada tahun 1969 oleh Depatemen Pertahanan Amerika telah merubah segala macam bentuk pengiriman pesan. Surat Elektronik (email) yang diperkenalkan oleh Raymon Samuel di era 1971 semakin membuat pengiriman pesan lebih efektif dan bisa menjawab kompleksitas dari materi pesan tersebut. Saat ini teknologi yang sedang tren adalah *chatting*, yaitu sebuah bentuk komunikasi secara langsung seperti bercakap cakap atau berkirim pesan secara real time menggunakan teks melalui jaringan internal, metode *chatting* dimulai dari *synchronous* sampai sekarang *Asynchronous JavaScript and XML (AJAX).*

Cara kerja *AJAX* untuk membuat *request* ke server secara *asynchronous* atau tanpa melakukan refresh halaman website yaitu dengan menggunakan *XML HTTP Request Object Javascript*, kemudian web server akan merenspon dan mengirimkan hasilnya melalui web browser. Salah satu kekurangan dari *AJAX* adalah beban dari HTTP header yang tidak efisien untuk pesan-pesan yang kecil.

AJAX sudah mampu menyajikan halaman web secara *real-time*, namun dengan metode *long polling* yang mengirimkan *request* secara terus menerus dikalikan jumlah pengguna yang banyak akan menimbulkan *server* sibuk bahkan *down*. Perkembangan teknologi yang sedang berkembang saat ini untuk komunikasi *real time* adalah *Websocket*. *Websocket* menggunakan komunikasi dua arah *(full-duplex)* melalui koneksi TCP tunggal [1].

Fokus penelitian ini adalah membandingkan kecepatan data *transfer* pada aplikasi *chatting* antara *Websocket* dan *AJAX*, untuk dijadikan perbandingan tercepat dan teroptimal diantara kedua teknologi tersebut.

2. LANDASAN TEORI

2.1. *Websocket*

Websocket adalah standar baru untuk komunikasi *full-duplex* (dua arah secara bersamaan) sehingga komunikasi yang terjadi antara *client* dan *server* lebih *real-time* [2].

2.2. *AJAX*

AJAX, singkatan dari "*Asynchronous JavaScript and XML*", merupakan metode suatu laman web menggunakan *Javascript* agar dapat menerima konten yang di *request* tanpa harus malakukan *refresh* pada halaman tersebut. Metode yang dapat digunakan *AJAX* adalah *polling* dan *long polling* [3].

2.3. *Long Polling*

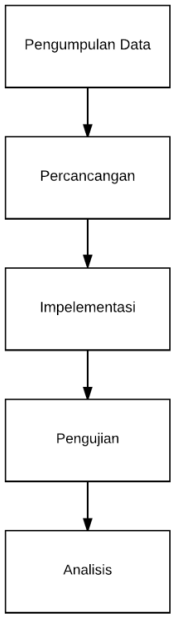
Metode ini yang diantaranya digunakan oleh teknologi *AJAX* dengan cara mengirimkan *request* secara terus menerus dari *client* ke *server*, hal tersebut menyebabkan *server* akan jadi sibuk dan rentan terkena serangan DDOS [4].

2.4. *Full Duplex*

Full Duplex adalah jenis komunikasi dua arah dimana kedua stasiun dapat melakukan transmisi secara simultan, keduanya bisa berkomunikasi, medium membawa dalam dua arah pada waktu yang sama [5].

3. METODOLOGI

Ada 5 tahapan dan metode yang akan di lakukan di antaranya: Pengumpulan Data, Perancangan, Implementasi, Pengujian dan Analisis yang dapat dilihat pada Gambar 1.



Gambar 1 Tahapan metodologi

3.1. Pengumpulan Data

Pengumpulan data dilakukan untuk mengumpulkan berbagai data dan informasi yang diperlukan dan berhubungan dengan penelitian. Tahapan-tahapan yang dilakukan pada pengumpulan data yaitu studi kepustakaan.

3.2. Perancangan

Tahapan perancangan yaitu pembuatan aplikasi yang akan dibuat, mulai dari kode *websocket* menggunakan php dan *javascript*, kode *AJAX* menggunakan PHP dan *javascript* dengan terintegrasi pada *database*, beserta hasil pembuatan aplikasinya.

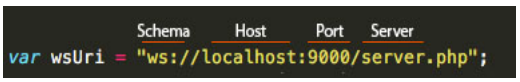
1. Perancangan *Websocket*

Merancang sebuah sistem *chatting* sederhana menggunakan teknologi *websocket* dan *PHP*, keseluruhan kode *Websocket* terdiri dari beberapa metode dan *event*. Gambar 2 adalah *source code* perancangan koneksi *websocket*.

```
websocket = new
WebSocket("ws://localhost:9000/serve
r.php");
websocket.onopen = function(evt) {
/* Perintah */ };
websocket.onclose = function(evt) {
/* Perintah */ };
websocket.onmessage = function(evt)
{ /* Perintah */ };
websocket.onerror = function(evt) {
/* Perintah */ };
websocket.send(message);
websocket.close();
```

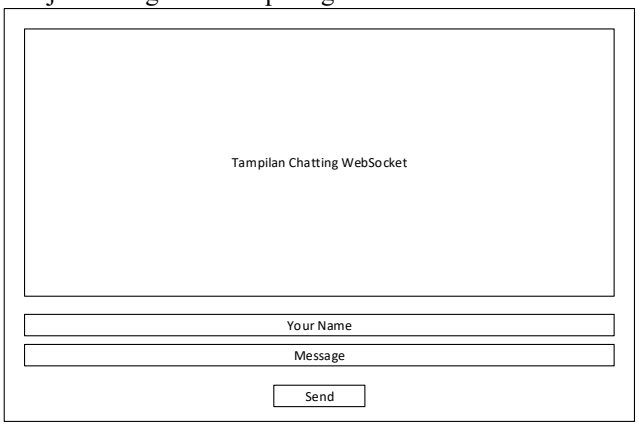
Gambar 2 Koneksi Websocket

Untuk membuka koneksi *socket* cukup memanggil *New Websocket(ws://SERVER URL)*, karena *websocket* menggunakan protokol yang berbeda untuk koneksi *ws://* daripada *http://* dan di ikuti oleh *host*, nomor *port*, dan *script server.php,* dengan skema koneksi *websocket* yang dapat dilihat pada gambar 3.



Gambar 3 Skema Pembuatan Koneksi *WebSocket*

Tepat setelah membuka koneksi perlu melampirkan beberapa *event handler* yang memberitahu tentang status *konektivitas*, kesalahan dan pesan masuk. Pembuatan aplikasi *chatting websocket* digabungkan dengan *library ajax.* Tahap berikutnya membuat sebuah html dan tampilan css yang *simple*. *WebSocket* yang berjalan secara permanen atau *realtime*, melakukan *handsaking websocket*, mengirim atau menerima data dari halaman obrolan dan menangani banyak klien. Setelah *handshaking*, klien sekarang dapat mengirim dan menerima pesan namun, pesan yang dikirim semuanya terenkripsi, dengan perintah *php server.php* maka, *server* berjalan dengan baik seperti gambar 4.



Gambar 4 *Interface Chatting Websocket*

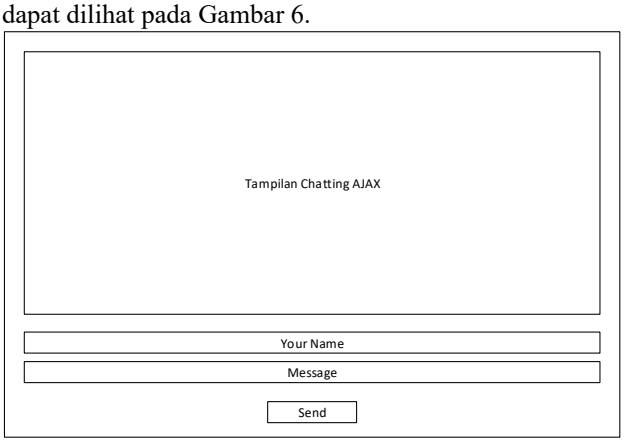
2. Perancangan *AJAX*

Teknologi *ajax* dalam *chatting* biasanya menggunakan *setInterval()* dari *JQuery* sehingga program akan me-*refresh* obrolan setiap milidetik, memuat data dari *database* yang dikembalikan ke dalam *element*, memperbarui *shout-box* dengan pesan baru yang ditambahkan.

```
// load Pesan Setiap 1000 milidetik
dari server.
load_data = {'fetch':1};
window.setInterval(function(){
$.post('content/refresh.php',
load_data, function(data) {
$('.message_box').html(data);
var scrolltoh =
$('.message_box')[0].scrollHeight;
$('.message_box').scrollTop(scrolltoh
);
});
}, 1000);
```

Gambar 5 Perancangan interval *Ajax*

Perancangan interval AJAX dapat dilihat pada Gambar 5 sedangkan rancangan tampilan aplikasi chatting dapat dilihat pada Gambar 6.



Gambar 6 *Interface Chatting Ajax*

3.3. Implementasi

Tahapan implementasi yaitu tahapan memasukan aplikasi chatting *websocket* dan *AJAX* yang sudah di rancang pada *server* dan installasi *docker* pada *server* dengan *container cadvisor, grafana, prometheus* dan *debian* sehingga bisa membandingkan aplikasi chatting mana yang lebih baik dalam segi kecepatan.

3.4. Pengujian

Tahapan pengujian akan dilakukan pada aplikasi chatting dengan menggunakan Mozilla Firefox selama 30 menit dengan perbandingan antara aplikasi chatting websocket dan ajax secara bersamaan dengan 4 user.

3.5. Analisis

Tahapan analisis yaitu membandingkan aplikasi *chatting* menggunakan *AJAX* dengan aplikasi *chatting* menggunakan *websocket* menggunakan *container cadvisor*, *promethius*, dan *grafana* dari aplikasi *server docker* dengan melihat kecepatan pengiriman, membandingkan penggunaan *bandwidth* pada *websocket* dan *AJAX,* membandingkan kinerja *websocket* dengan *AJAX*. Hasil dari perbandingan yang dilakukan adalah menentukan aplikasi yang paling lebih cepat dalam pengiriman pesan dan hemat dalam penggunaan *bandwith*.

4. HASIL DAN PEMBAHASAN

4.1. Implementasi

Tahapan implementasi penulis menjelaskan installasi menggunakan aplikasi *virtualisasi* modern pada *server centos 7* yaitu menggunakan *docker* yang didalamnya terinstall *container cadvisor, grafana, prometheus* dan *container opriting system* debian *jessie* yang sudah terinstall untuk kebutuhan aplikasi chatting dengan teknologi *websocket* maupun *ajax* di dalam *container* tersebut berjalan sesuai yang diharapkan sehingga dapat membandingan tekonologi *ajax* dan *websocket* dalam segi kecepatan maupun penggunaan memory CPU, berikut tahapan installasi dan penerapannya.

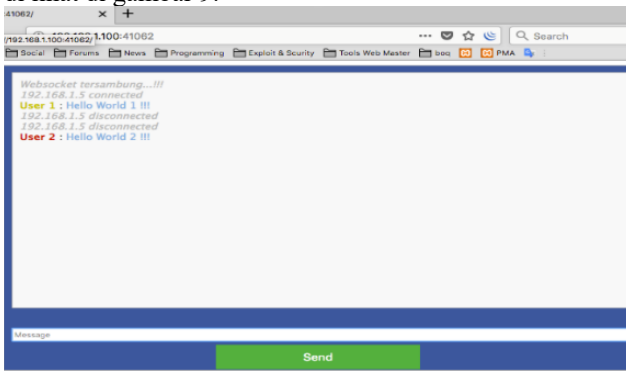
1. *Install Docker*
   Install docker pada *centos 7* dengan perintah *curl -fsSL https://get.docker.com/ | sh.*

Gambar 7 Installasi *Docker Engine*

2. *Install Container Debian* untuk *WebSocket*
   Menjalankan aplikasi websocket dengan perintah *php –q server.php* di *container debian*

Gambar 8 Menjalankan *websocket*

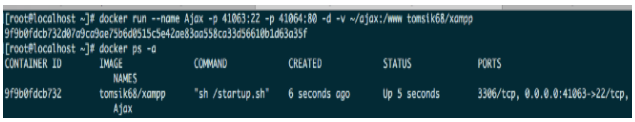
Hasil dari menerapkan teknologi *weboscket* chatting dengan *container debian jessie* yang sedang berjalan bisa di lihat di gambar 9.



Gambar 9 Mengirim aplikasi *websocket* ke *server*

3. *Install Container Debian* untuk *AJAX*

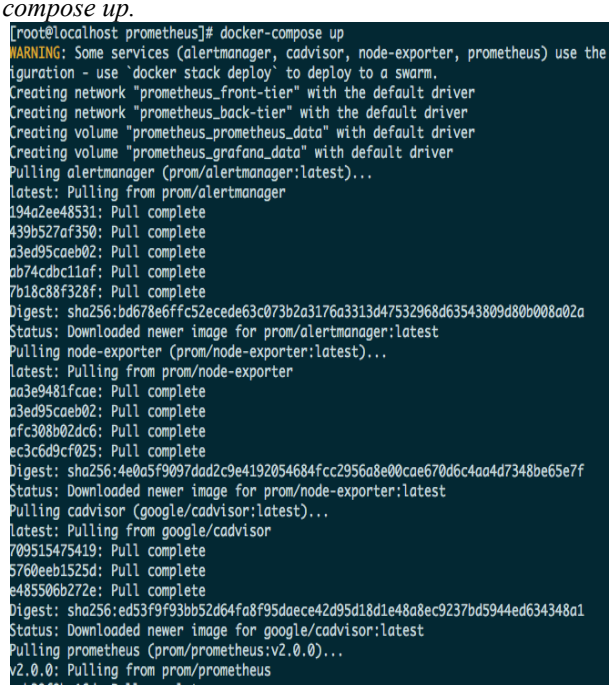
Membuat sebuah *container debian* (Gambar 10) dengan perintah *docker run --name Ajax -p 41063:22 -p 41064:80 -d -v ~/ajax:/www tomsik68/xampp*



Gambar 10 Membuat *Container debian*

4. *Install Container Cadvisor, Prometheus* Dan *Grafana*

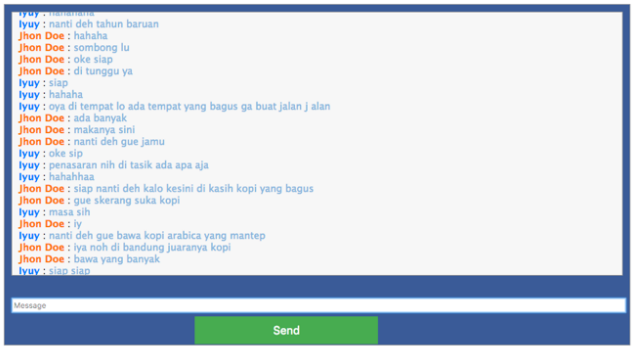
Install *docker-compose* dengan masuk ke dalam folder *Prometheus* lalu mengetikan perintah *docker-compose up.*



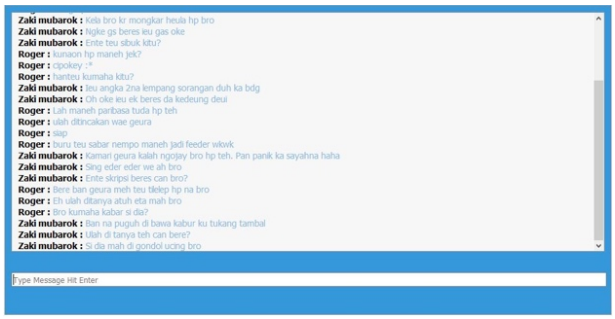
Gambar 11 Mendownload *image cadvisor, Prometheus dan grafana*

4.2. Pengujian

Tahapan pengujian akan dilakukan pada aplikasi *chatting* dengan menggunakan Mozilla Firefox selama 30 menit dengan perbandingan antara aplikasi *chatting websocket* dan *ajax* secara bersamaan dengan 4 *user*. 2 *user* aplikasi *chatting* dengan *websocket*, 2 *user* aplikasi *chatting* dengan *ajax* selama 30 menit, dengan tujuan akhir membandingkan aplikasi mana yang lebih baik diantara keduanya dan dapat dilihat pada gambar 12 dan 13.



Gambar 12 Aktifitas *chatting websocket* saat di bandingkan



Gambar 13 Aktifitas *chatting ajax* saat di bandingkan

4.3. Analisis

Tahapan analisis ini menjelaskan perbandingkan dari *websocket* dengan *ajax* pada *transmit* data *network* dan *receive* data *network*, hasil dari *chatting* pada tahapan pengujian bisa dilihat pada tabel 1.

TABEL 1. TRANSMIT DATA NETWORK

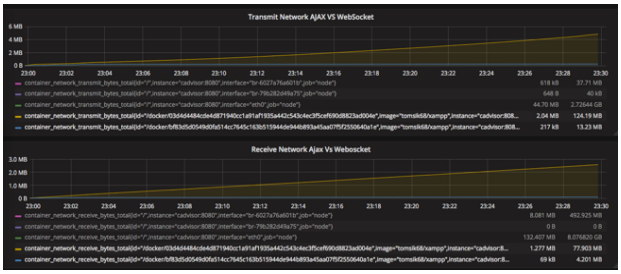
Menit/ Percobaan	Ajax	Websocket
1	246 Kb	19 Kb
2	312 Kb	116 Kb
3	473 Kb	207 KB
4	553 Kb	210 Kb
5	635 Kb	214 Kb
6	724 Kb	215 Kb
7	811 Kb	216 Kb
8	917 Kb	218 Kb
9	1.023 Mb	219 Kb
10	1.138 Mb	221 Kb
11	1.261 Mb	224 Kb
12	1.392 Mb	225 Kb
13	1.540 Mb	228 Kb
14	1.681 Mb	231 Kb
15	1.834 Mb	231 Kb
16	1.998 Mb	233 Kb
17	2.164 Mb	236 Kb
18	2.344 Mb	238 Kb
19	2.529 Mb	241 Kb
20	2.708 Mb	242 Kb
21	2.884 Mb	245 Kb
22	3.076 Mb	247 Kb
23	3.269 Mb	248 Kb
24	3.465 Mb	249 Kb
25	3.671 Mb	252 Kb
26	3.885 Mb	252 Kb
27	4.106 Mb	254 Kb
28	4.337 Mb	255 Kb
29	4.632 Mb	257 Kb
30	4.887 Mb	258 Kb
Rata-rata	2.04 Mb	217 Kb
Total	124.19 Mb	13.23 Mb

Pada tabel 1 diatas hasil persentase dari *transmit data network* ajax adalah 90,37% dan sementara hasil persentase *websocket* adalah 9,63%. Sehingga dari data yang telah dibandingkan, *websocket* lebih efisien dalam penggunaan *transmit* data *network* daripada ajax.

TABEL 2. RECEIVE DATA NETWORK

Menit/ Percobaan	Ajax	Websocket
1	161 Kb	39 Kb
2	206 Kb	41 Kb
3	299 Kb	50 KB
4	384 Kb	53 Kb
5	463 Kb	56 Kb
6	543 Kb	57 Kb
7	618 Kb	58 Kb
8	701 Kb	60 Kb
9	783 Kb	61 Kb
10	866 Kb	62 Kb
11	949 Kb	64 Kb
12	1.034 Mb	66 Kb
13	1.121 Mb	68 Kb
14	1.200 Mb	71 Kb
15	1.283 Mb	71 Kb
16	1.370 Mb	72 Kb
17	1.457 Mb	75 Kb
18	1.542 Mb	77 Kb
19	1.628 Mb	79 Kb
20	1.719 Mb	80 Kb
21	1.797 Mb	83 Kb
22	1.882 Mb	85 Kb
23	1.971 Mb	86 Kb
24	2.057 Mb	87 Kb
25	2.103 Mb	89 Kb
26	2.280 Mb	90 Kb
27	2.327 Mb	91 Kb
28	2.417 Mb	92 Kb
29	2.511 Mb	94 Kb
30	2.595 Mb	94 Kb
Rata-rata	1.277 Mb	69 Kb
Total	77.903 Mb	4.201 Mb

Pada tabel 2 diatas hasil persentase dari *receive data network* ajax adalah 94,88% dan *websocket* 5,12%, Hasil yang ditunjukkan dari *docker grafana* dapat dilihat pada gambar 14.



Gambar 14. Grafik docker Grafana

D. Kelebihan dan Kekurangan

Kelebihan pada sistem yang telah dibangun sebagai berikut:

1. Perbandingan ajax telah dilakukan dengan hasil persentase *transmit* data network 90,37% dan websocket 9,63% sementara persentase *receive* data network ajax 94,88% dan websocket 5,12%.
2. *Websocket* merupakan standar protokol jaringan yang dapat melayani standar protokol lain di atasnya

Kekurangan pada sistem yang telah dibangun: Data hasil dari *docker grafana* dan *prometheus* tidak seterusnya konsisten.

5. KESIMPULAN

1. *Websocket* dapat diimplementasikan pada aplikasi *chatting* dan mampu menghemat *transmit* data *network* serta *receive* data *network* dari ajax.
2. Telah berhasil menguji kecepatan *transmit* dan *receive* data *network* antara *websocket* dengan *ajax* dalam aplikasi *chatting*.
3. Perbandingan *ajax* telah dilakukan dengan hasil persentase *transmit* data *network* 90,37% dan websocket 9,63% sementara persentase *receive* data *network ajax* 94,88% dan websocket 5,12% .
4. *Weboscket* secara teknis memungkinkan *server* untuk mendorong data kepada klien yang terhubung dengan protokol komunikasi dua arah yang dapat digunakan oleh *browser* dibandingkan dengan *ajax* di karnakan *ajax* mengambil data web secara *background* dan memperbarui isi sebagian halaman.

Sebaiknya dalam penelitian selanjutnya dilakukan dengan aplikasi perbadingan khusus untuk *benchmark* dan *comparison* baik di dalam lalu *lintas latency* maupun penggunaan *memory* dan CPU.

DAFTAR PUSTAKA

[1] H. Husen, A. Rahmatulloh and H. Sulastri, "Implementasi Komunikasi Full Duplex Menggunakan Sistem Informasi Pengelolaan Anggaran Universitas ABC," *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer,* vol. 9, no. 1, pp. 597-606, 1 4 2018.

[2] Q. Liu and X. Sun, "Research of Web Real-Time Communication Based on Web Socket," *Int. J. Communications, Network and System Sciences,* vol. 5, pp. 797-801, 2012.

[3] I. Hidayat, "Perbedaan Websocket dengan AJax," 5 12 2015. [Online]. Available: http://izalhidayat.student.telkomuniversity.ac.id/perbedaan-websocket-dengan-ajax/. [Accessed 19 12 2017].

[4] R. Appelqvist and O. Örnmyr, "Performance comparison of XHR polling, Long polling, Server sent events and Websockets," *Performance comparison of XHR polling, Long polling, Server sent events and Websockets.*

[5] T. C. SCIENCE, "Simplex, Half Duplex, Full Duplex," [Online]. Available: https://teachcomputerscience.com/simplex-half-duplex-full-duplex/. [Accessed 01 2018].

BIODATA PENULIS

Asep Rizki Maulana
Mahasiswa Teknik Informatika, Fakultas Teknik, Universitas Siliwangi. Selain kegiatan dikampus berperan aktif juga dalam komunitas web developer dan hacker Indonesia.

Alam Rahmatulloh
Dosen Teknik Informatika, Fakultas Teknik, Universitas Siliwangi, berperan aktif dalam bidang web, IoT, keamanan informasi, mobile dan android.